INFORMATICS PROFESSIONALS. LEADING THE WAY.

Using iSyntax Whole Slide Image Format in Research/ML

Making Sense of Data with Visualizations

S37

Alexandr Virodov

University of Kentucky

alexandr.virodov@uky.edu

Disclosure

I have no relevant relationships with commercial interests to disclose.

Learning Objectives

After participating in this session the learner should be better able to:

- Understand handling of Philips iSyntax whole slide images
- Learn about OpenSlide use for Philips iSyntax

Introduction

- Pathology Images
  - Very large, 100k x 100k pixels typical
  - Multiple scales
  - Applications: Viewing, Machine Learning
- Libraries & API to access pathology images
  - OpenSlide, BioFormats, Cucim
  - Formats supported: .svs, .tif, others. Not iSyntax.
- Philips iSyntax
  - First FDA-approved scanner, widely used
  - Pyramidal, Wavelet-based format. Similar to Jpeg2000 but faster due to different coding.
  - Closed source, SDK difficult to obtain, individual licensing

Usecases

- Viewing
  - Integration with Digital Slide Archive viewer
  - Random access on multiple levels, single/few slides, small throughput
- ML – small batch
  - MIL (Multiple Instance Learning), other tile-based sampling algorithms
  - Random access on single level, many slides
- ML – sequential read
  - Tissue detection
  - Cell detection
  - Sequential access on single level

Our Solution

Pre-existing: open-source iSyntax decoder in Slidescape viewer [2].

Our work:

- Integration into OpenSlide
  - We evaluated BioFormats, CuCim as alternatives
  - OpenSlide implemented in C, matching Slidescape's C/C++
  - Backend for QuPath, MONAI, DSA
- Shared caching of tiles, optimized for ML usecase

[2] Pieter Valkema, Slidescape. https://github.com/amspath/slidescape

Alternatives

- Conversion
  - Slow (~3 min / slide, 10k slides = 20.8 days to convert), not robust – hangs
  - Requires choice of tif jpeg compression level. At 80%, quality degrades, at 100% size doubles
  - Duplication of data

- Wrapping the SDK
  - Requires obtaining SDK
  - SDK is limited to specific versions of Linux / Python
  - OpenPhi – Python library with same functions as OpenSlide, but would require code changes in e.g. MONAI

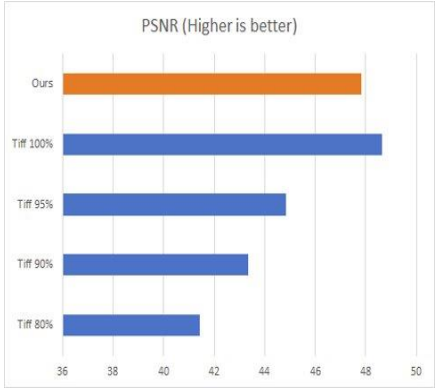
Evaluation - Storage

- Storage/Quality tradeoff
- Pyramidal overhead
- (Conversion is single-process)

Philips SDK Conversion (Tiff Jpeg Quality)	80%	90%	95%	100%
Time to convert 10 slides	32m 9s	29m 48s	31m 58s	40m 26s
Additional storage for 10 converted slides (iSyntax size: 9.4 Gb)	4.7 Gb	6.9 Gb	12 Gb	27 Gb
Additional storage as percent of iSyntax size	50%	73%	128%	287%

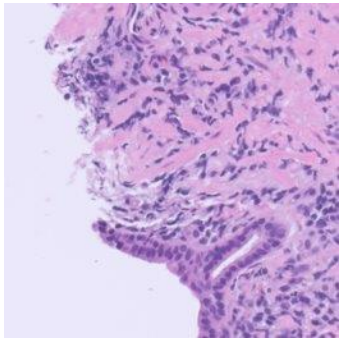
Evaluation - Quality

- PSNR
- Visual comparison
- Sources of error
  - Not bit-precise decoding
  - Occasional blue pixels – YCoCb out-of-bounds

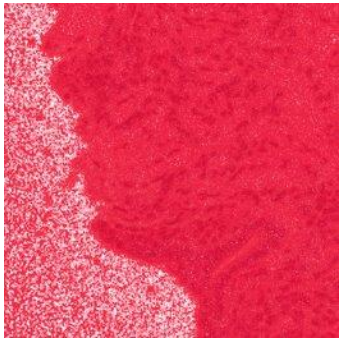


PSNR (Higher is better)	
Tiff 80%	41.43
Tiff 90%	43.35
Tiff 95%	44.85
Tiff 100%	48.65
Ours	47.83

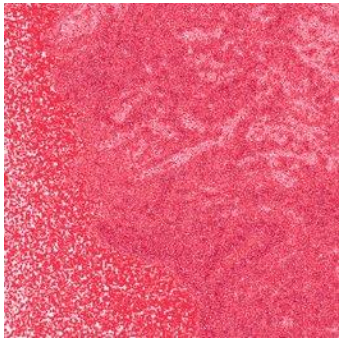
(error image & PSNR are computed by imagemagick `magick compare -metric PSNR` compared to "Test Region" extracted as .png using Philips SDK)



Test Region



Error Tiff 95%



Error Ours

Evaluation – Speed & Memory

- Last level is very slow, but not always needed
- Bottleneck may not be in reading
  - Monai-MIL – bottleneck is GPU
  - Viewing – bottleneck is human viewing throughput
- (Handling of sparse tiles makes direct comparison hard)

Access 100 slides	Time Tiff 95% (s)	Memory Tiff 95% (Mb)	Time Ours (s)	Memory Ours (Mb)	Time Ours / Tiff
Full Sequential read of last level	6.74 hrs)	3,311	13.5hrs	7,040	2.01
Random single tile read of last level	0.24	210	3.94	7,057	16.55
Random single tile read of middle level	0.22	158	1.99	6,919	8.99
Random 100 tile read of last level	18.75	3,311	235.32	10,181	12.55
Random 100 tile read of middle level	11.71	217	5.33	7,048	0.45
Monai-MIL [7]	4081	161,698	4810	95,946	1.17

Limitations

Study limitations:

- Single process for evaluating Philips SDK conversion. Horizontal scaling is easy.
- Handling of sparse tiles makes direct comparison hard

Conclusions

- Our solution is a good alternative to conversion if the reading speed is not the bottleneck
  - Viewing
  - Some ML loads
  - Additional caching layers present
  - Random access
- Our solution avoids conversion overhead (time, storage)
- Our solution does not require code changes for current OpenSlide users

Future work

- Reducing per-slide memory consumption by evicting internal structures
- Better multithreading support
- Runtime performance improvements
- Investigating decoding errors (low-bit noise, YCoCb out-of-bounds)
- Evaluating integration into Bioformats, CuCim

Thank you!

Email me at:
alexandr.virodov@uky.edu